AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning at page 1, line 5, as follows:

BACKGROUND ~~OF THE INVENTION~~ AND SUMMARY

Please amend the paragraph beginning at page 3, line 4, as follows:

There are systematic and non-systematic FEC channel encoders. A systematic encoder creates code-words by appending redundant bits, sometimes called parity bits, to a block of information bits. A non-systematic encoder maps a set of information bits into a completely different set of coded bits. One class of powerful systematic codes is turbo codes. A turbo encoder uses at least two recursive convolutional encoders separated by an interleaver. Information bits are sent directly to the first encoder, and via the interleaver, to the second encoder. The turbo encoded codewords ~~includes~~ include the original information bits (i.e., the systematic bits) and additional parity bits from the first and second encoders. Because the systematic bits are used in both the first and second encoders, the systematic bits are more important to the decoder at the receiver than the parity bits. Indeed, if the systematic bits are lost, it may be impossible for the decoder to correctly decode the data packet even if a substantial number of parity bits are subsequently received.

Please amend the paragraph beginning at page 7, line 12, as follows:

DETAILED DESCRIPTION ~~OF THE INVENTION~~

Please amend the paragraph beginning at page 9, line 13, as follows:

Fig. 4 shows a hybrid ARQ with partial incremental redundancy (IR) packet combining in accordance with one example, non-limiting embodiment of the invention. Here, the properties of both types of combining illustrated in Fig. 2 and Fig. 3 are used. Each time the transmitter receives an NACK signal from the receiver, it retransmits the same bits from S_1, but a new set of parity bits $P_{1,k}$ ($k = 2, 3, \ldots$) is selected for each retransmission. The receiver combines the different transmissions of the bits in S_1 in order to make them more reliable, and the FEC decoder uses the bits in the different parity bit sets $P_{1,k}$ ($k = 2, 3, \ldots$), to improve the error correction ability of the FEC decoder. Eventually the packet is correctly decoded, and the receiver sends an ACK signal back to the transmitter.

Please amend the paragraph beginning at page 10, line 14, as follows:

In contrast, the hybrid ARQ, partial IR scheme depicted in Fig. 4 solves this lost packet problem. As shown in Fig. 6, ~~the~~ that hybrid ARQ, partial IR scheme is more robust than the hybrid ARQ full IR scheme because it overcomes the situation when the original transmission is lost or substantially corrupted. The retransmissions in this scheme include the bits in S_1, and for each retransmission, a new set of parity bits $P_{1,k}$ ($k = 2, 3, \ldots$). Since the decoder receives the important bits, e.g., S_1, in the retransmission, the packet can be decoded. However, because each retransmission contains relatively few redundancy bits, the corresponding coding gain is reduced as compared to that of a full IR scheme where the retransmissions include only redundancy

bits. Thus, performance and efficiency are traded off in the partial IR scheme for the robustness in handling retransmissions of lost packets.

Please amend the paragraph beginning at page 10, line 25, as follows:

Another solution to the substantially lost/corrupted original packet transmission problem is depicted in ~~another~~ an example, non-limiting embodiment of the ~~present invention~~ shown in Fig. 7. If the receiver does not receive a data packet or the received data packet is severely corrupted, the receiver sends a LOST signal, (i.e., not a NACK, nor an ACK signal), back to the transmitter. The LOST signal triggers the transmitter to include the more important bits in the group S_1 in the following retransmission, and preferably (though not necessarily), bits from the less important group P_1, e.g., $P_{1,1}$. On the other hand, if the receiver decides that the received data packet is useful, but errors are still detected in the decoder output, the receiver sends a NACK signal back to the transmitter. The NACK signal triggers the transmitter to send a retransmission including bits from the less important group P_1, e.g., $P_{1,2}$, without having to include the more important bits, e.g., S_1,. In this way, each retransmission enjoys a greater coding gain than the solution in Fig. 4. Once the data packet is successfully decoded by the receiver, an ACK signal is sent back to the transmitter triggering transmission of the next data packet. One interpretation of the NACK and LOST signals is that the receiver transmits a NACK in order to ask for a retransmission containing mainly additional less important bits and a LOST signal to ask for a retransmission containing mainly the important bits.

Please amend the paragraph beginning at page 12, line 3, as follows:

In the example embodiments of Fig. 4 and Fig. 6, the receiver does not always need three feedback responses (ACK/NACK/LOST). As long as the receiver can keep track of which ~~of the~~ less important bits ~~that~~ are included in a retransmission, it is sufficient to use only two feedback responses (ACK/NACK). However, there are situations when using three feedback responses may still be beneficial even for the partial IR combining scheme. For example, if the receiver completely misses the original transmission or one of the retransmission, it might loose count ~~on~~ of the retransmissions. Thus, if the receiver detects that the set of less important bits in a retransmission does not correspond to what the receiver ~~had~~ expected, it may send a LOST signal back to the transmitter which triggers the transmitter to start all over again with the original transmission. Alternatively, the receiver may not have enough buffer memory in the combiner buffer to store any additional less important bits. By sending a LOST signal to the transmitter, the following retransmission will contain only less important bits that has previously been transmitted, and which does not require any new buffer memory.